UCRL-ID-115099 Rev. 2

Livermore Computing Production Control System Product Description Rev. II

R.R. Wood

October 1994

Lawrence Livermore National Laboratory

Livermore Computing Production Control System

Acknowledgments

Designing and implementing the PCS product has been challenging and rewarding. Solving the problems and issues involved has been an interesting experience and the team put together to develop requirements and to implement them has worked together very well. I sincerely thank the many people who have contributed large amounts of time and energy to the development of the PCS.

Bob Cooper, manager LC Client Services, laid out customer requirements and was very helpful in leading customer discussions. Barbara Atkinson, LC project manager, tirelessly made sure that “bases were covered.”

Without substantial input from the user community, the PCS could not have been written to solve real problems experienced by coordinators. Doug Dickson, Nancy Alexander, Ken Giallanza and Lynn Groves, customer coordinators, attended meetings held once per week during a period that covered two years. At these meetings, they laid out customer requirements comprehensively and graciously showed the flexibility needed in trading off desires against feasibility, efficiency, etc.

Don von Buskirk, project member, brought twenty seven years operating systems programming experience to the development of the Production Workload Scheduler. Phil Eckert, has many years experience in managing resources on production machines. He was and is a “sounding board” for many of the design ideas. (He has kept a few less-than-perfect ideas from seeing the “light of day!”) Phil also wrote significant portions of the RAC system and injected humor at the nicest times--when it was badly needed.

Table of Contents

Introduction

The purpose of this document is to introduce and describe the Livermore Computing Production Control System, commonly called the PCS. The intended audiences for this document are system administrators and resource managers of computer systems who intend to use or evaluate the PCS.

Mission

The Production Control System (PCS) provides:

. basic data reporting mechanisms required for project level, near-real time accounting systems.

. means for customers of UNIX based production systems to be allocated resources according to an organizational budget. Customers are then able to control their users' access to resources and to control the rate of delivery of resources.

. mechanisms for automated delivery of resources to all production being managed by the system.

. graceful degradation in service to prevent the overuse of the machine where not authorized.

. proactive delivery of resources to organizations that are behind in consumption of resources to the extend possible through the use of the underlying batch system.

The PCS is NOT a CPU scheduler; rather, it relies on a standard kernel scheduler to perform this function. It is not a batch system. However, it may be regarded as a policy enforcement module which controls the functioning of a batch system. Finally, it does not do process level, process termination accounting.

While the PCS is not a memory or CPU scheduler, it does adapt the production demand on the machine to present an efficiently manageable workload to the kernel's memory and CPU schedulers. It monitors memory load, swap device load, idle time, etc. to keep resource demands within bounds that are configurable by site administrators.

Historical Perspective

In 1990, Livermore Computing committed itself to convert its production platforms to UNIX based systems. This was a radical change for Livermore Computing in that we had developed and come to rely on elaborate facilities in our earlier, proprietary systems to help the lab's scientists productively use the machines and to accurately account for their use to government oversight agencies.

UNIX was found to have extremely weak means by which the machine's resources could be allocated, controlled and delivered to organizations and their users. Accounting also is weak, being generally limited to process level accounting and then only at process termination. Thus, resources can only be allocated to users and can only be accounted for after they are used. No cognizance can be taken of organizational structure or projects. Large computing centers frequently have thousands of users working on hundreds of projects. These users and the projects are funded by several organizations with varying ability or willingness to pay for the computer services provided. With only typical UNIX tools, the appropriate delivery of resources to the correct organizations, projects, tasks and users requires continual human intervention. Denying service to users who have consumed their allocated resources is crude: administrators can beg users to log off, can "nice" undesirable processes or can disable logins. We need automatic, less draconian means to degrade service to users .

The PCS project, begun in 1991 and in operation since October 1992, addresses UNIX shortcomings in the areas of project and organizational level accounting and production control on the machines.

Architectural Overview

The PCS is used to allocate, control and assure the proper delivery of resources on production capacity computer systems. It is also an integral part in collecting and reporting resource usage and for charging for resource usage. There are two major components of the PCS. They are the Resource Allocation & Control system (RAC) and the Production Workload Scheduler (PWS).

The PCS system requires the RAC subsystem to run, but may be configured to run without the PWS. The PWS may not, however, run unless the RAC system is also running.

Resource Allocation & Control system (RAC)

Generally, the RAC system is used to create and manage recharge accounts, to create and manage allocation pools and to create and manage user allocations within the allocation pools. A recharge account should not be confused with a user “login” account. So that the term “group” not continue to be overused, the PCS has borrowed another term to mean an allocation pool or group. This term is “bank”.

As resources are consumed on the machine the RAC system associates those resources with the users who are consuming them. Then the resources are associated with a bank and a recharge account. The user and the bank are debited the resources used and an accounting report is made for the purpose of charging the account. All of this is done in near-real time. If the RAC system determines that a user, recharge account or bank has consumed as much of the resources as has been permitted, the RAC system takes steps to prohibit further resource consumption by that user, recharge account or bank. There are facilities by which users may specify the banks and accounts to be charged for their production.

recharge accounts

A recharge account, or simply account, is essentially a credit that represents an amount of usable resources (which may be unlimited). Users and banks may be permitted to charge an account for resources used. Some users, called account coordinators, may be permitted to manage the account. That is, account coordinators may grant and deny other users access to an account. Accounts are independent from each other; that is, accounts have no sub-accounts.

The primary purposes of accounts are 1) as a mechanism by which budgetary charges may be determined for organizations whose personnel use the computer and 2) a “one stop” limit on resource accessibility to users.

domains and partitions

Eventually, a site will be able to specify several heterogeneous hosts as a single domain to be managed by the PCS. Currently, a domain definition is restricted to contain only one host. Within a domain, the nodes that make up a host may be partitioned.

banks

A primary purpose of the banking system is to provide a mechanism by which the rate of delivery of allocated resources is managed. Also, the PWS uses the banking system to prioritize and manage production on the machine. This is further described in the PWS section below.

A bank represents a resource pool available to sub-banks and users who are permitted access to the bank. As implied, banks exist in a hierarchical structure. There is one “root” bank which “owns” all resources on a machine. Resources of the root bank are apportioned to its sub-banks. The resources available to each bank in turn may also be apportioned among its sub-banks. There is no limit to the depth of the hierarchy. This hierarchy is exemplified in Figure 1. In this figure, banks B and C are sub-banks of and draw their resources directly from bank A. Bank D is a sub-bank of and draws its resources from bank B. Banks G and H are sub-banks of and draw their resources from bank D. The boxes labeled U1 through U4 represent user allocations to banks G, H, E and F, respectively. A bank may have both user allocations and sub-banks. (This is not shown in the figure.)

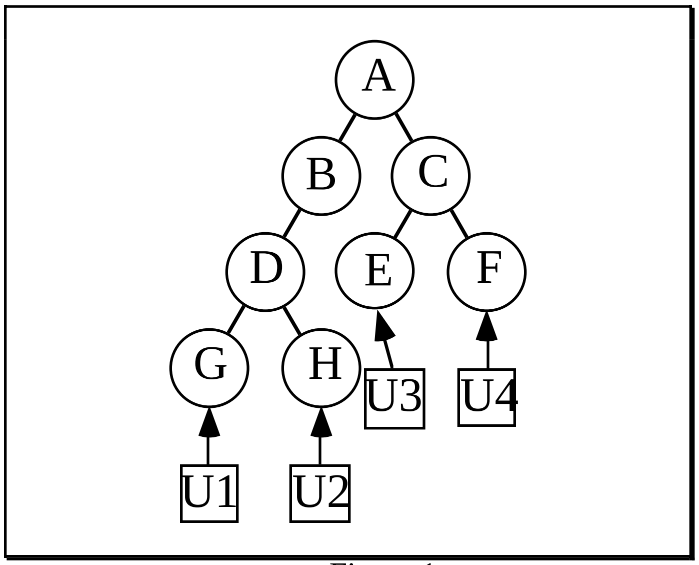

Figure 1
Banks are organized in a hierarchy. The root bank is the top level bank. There is no limit on the depth of the hierarchy. User allocations may be made at any level. A user may have allocations to more than one bank. Any number of users may have allocations to one bank.

Bank resources are allocated on a partition basis. Some users, called bank coordinators, may create and destroy sub-banks and may grant and deny other users access to a bank. Bank coordinator designees may grant and deny users access to a bank, but may not create or destroy sub-banks. The authority of coordinators extend from the highest level bank at which they are named coordinator through out that bank’s sub-tree. The same is true of designees.

user allocations

Users are permitted access to a part or all of bank resources through a user allocation.

Production Workload Scheduler (PWS)

The Production Workload Scheduler schedules production on the machine. Production requirements are made known to the PWS in the form of batch requests. When the PWS is installed, users do not submit their requests directly to the batch system, but rather submit them to the PWS which then submits them to the batch system.

One important function of the PWS is to keep the machine busy without permitting it to be so busy that it begins thrashing. Interactive work on the machine is not scheduled by the PWS. However, the PWS does track the resource load presented by interactive usage and adjusts the amount of production to "load level" the machine.

At any point, there is a set of production requests being managed by the PWS. This set is called the production workload. Requests in this workload are prioritized according to rules and allocations laid out by system administrators and coordinators. High priority requests are permitted to run insofar as the machine is not overloaded.

When users submit a batch request, they must specify the bank from which resources are to be drawn and the account to be charged for the request's resources.

When the PWS determines that a production request should be run, it uses a "tree walk" algorithm to decide which production request to run. That is:

a. When the PWS determines that a new request should be scheduled, a search begins at the root bank. (The root bank is chosen as the "current" bank.)
b. If one or more runnable requests is drawing from a user allocation to the current bank, one of them is scheduled and the algorithm is complete.
c. A prioritized list of child banks of the current bank is constructed. (See bank prioritization below.)
d. For each bank in the list continue at step b.
e. Indicate that no request can be scheduled from the sub-tree.

In other words, a recursive search is made through the tree. At each level, sibling bank's sub-trees are searched in the order of the bank's relative priority. The tree walk is exemplified in Figure 2 on page 6. If the tree is represented with highest priority sibling banks pictured from left to right, then the path taken in the search for a job is as shown in the figure. Mathematically, this is called a preorder search. The search is stopped as soon as an executable job is found.

bank prioritization

The PWS uses a mechanism called adaptive scheduling to prioritize a set of sibling banks. Simply stated, to schedule a set of sibling banks adaptively is to schedule from the bank (or its sub-tree) which has the highest percentage of its allocated time not yet delivered in the current shift.

Each bank has associated with it a scheduling priority. This priority is a non-negative, floating-point number. The scheduling priority is a multiplier or accelerator on the bank's raw adaptive scheduling priority.

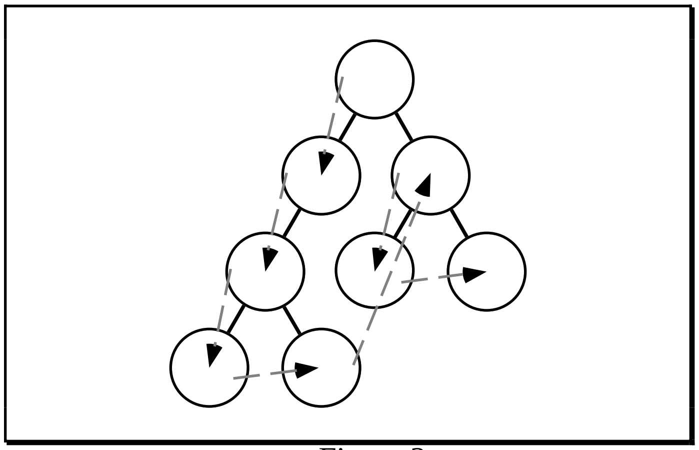

Figure 2
A “tree walk” is performed in the search for a request to run. Among a set of sibling banks, the bank with the lowest percentage of used time and its sub-tree is searched first.

request prioritization

Each request has associated with it several priority attributes. They are coordinator priority, intra-bank priority, also called the group priority and individual priority. These attributes establish a multi-tiered scheme used to prioritize requests drawing from the same bank. This prioritization scheme is discussed in detail in the configurability section.

exceptional scheduling

Some requests may be declared to be short production by a user. Requests with this attributes are scheduled on demand (as if they were interactive). Short production is discussed in more detail in the policy mechanisms section.

Requirements

The PCS requires an underlying batch processing system. The ability to checkpoint batch jobs is highly recommended where feasible, but not required.

There is also an administrative requirement. The center that intends to run PCS efficiently must put fairly strict limits on the interactive usage of the machines being controlled. For example, an interactive per process time limit of 10 CPU minutes and a per process memory limit of ~125MBytes on a Cray YMP 8/128 has been found sufficiently strict to permit the efficient functioning of PCS on that platform. At the same time, users should be encouraged to run their production via the batch system by removing time and memory limitations through the batch system to the extent possible. For instance, on the same YMP 8/128 mentioned above, a time limit of unlimited and a standard memory limit of ~656MBytes and exceptional memory limits as high as available memory has been found to be enough of a carrot to entice users away from using the machine interactively for production purposes.

Policy Mechanisms

There are several tunable parameters or “knobs” at the disposal of system administrators and coordinators that can be used to tailor the PCS system to a wide range of computing environments. These policy mechanisms are described in this section.

Bank Structure & Bank Prioritization

If two requests are drawing from two banks where one bank is the parent of the other, the request that is drawing from the parent bank will be delivered time before the other. A site can use this “prioritization by hierarchy” feature to manage the priorities of various production units. For example, a site may wish to have a “flat” structure between banks to assure that no production is considered to be higher priority than any other. Some organizations may have sub-units and may wish to prioritize these sub-units differentially. This feature accommodates both schemes simultaneously.

Scheduling Priority

Among a set of sibling banks, production is scheduled from them (and their sub-trees) adaptively. This means that when the percentage of already delivered resources from one bank falls behind that of its sibling, the PWS will favor the bank that is behind over its sibling banks.

Each bank has a scheduling priority which acts as a multiplier on the priority of a bank as determined by the adaptive scheduling algorithm. The effect of a high scheduling priority is to accelerate the delivery of resources to a bank until its allocated resources are depleted. Thereafter, it decelerates the delivery of resources (at low priority) until the bank resources are replenished. The effect of a low scheduling priority is to decelerate delivery (at high priority) of resources to a bank until its allocated resources are depleted. Thereafter, it accelerates the delivery of resources (at low priority) to a bank until the bank resources are replenished.

Simply stated, banks with high scheduling priority are favored until their allocated resources are delivered but it is more difficult for them to get shared time. (See the Time Sharing among Banks feature below.)

Over and Under Allocation

Assume bank **P** has been allocated 100 minutes of time and that it has child banks **A**, **B** and **C**. A coordinator for bank **P** is free to under allocate, to exactly allocate or to over allocate the time of bank **P** to its sub-banks.

over allocation

Over allocation implies competition for time. Assume the coordinator allocated 40 minutes each to banks **A**, **B** and **C**. This would imply that each bank would have available to it 20 minutes of non-contested time, but that they would have to compete for the remainder of bank **P**’s time. (Uncontested time for a bank is its parent’s time minus the sum of all its sibling banks’ times or zero if this calculation is negative.) The problem with over allocation is that the people drawing from bank **A** may not be ready to run production while in the meantime the people from banks **B** and **C** are using it. This would cause the people drawing from bank **A** to lose their time. Under allocation and exact allocation prevent this.

<u>under allocation</u>

Under allocation implies no competition for time but does imply reservation of time for the parent bank. Assume the coordinator allocated 20 minutes each to banks **A**, **B** and **C**. This would imply that each bank would have available to it 20 minutes each of non-contested time, but that the remaining 40 minutes of bank **P**'s time is (nominally) unavailable to any of them. The problem of under allocation is that time can not be used because it has not been allocated. The Time Sharing among Banks feature described below helps to alleviate this problem.

<u>exact allocation</u>

Exact allocation implies no competition for time and implies no reservation of time for the parent bank. Assume the coordinator allocated 33 minutes and 20 seconds each to banks **A**, **B** and **C**. This would imply that each bank would have available $33^1/_3$ minutes each of non-contested time, and that all of bank **P**'s time is available to its sub-banks. The problem with exact allocation is that if bank **A** does not use its time, that time is, nevertheless, not available to banks **B** or **C**. This is alleviated by the Time Sharing among Banks feature described below.

Time Sharing among Banks

Any bank can be marked as a "time sharing" bank. The scope of time sharing for a bank is its entire sub-tree except where prevented from sharing by the exclusion feature described below. The implication of sharing time is that all time available to a bank, but not allocated to a sub-bank is delivered to the sub-bank at a very low priority.

From the above examples, assume the under allocation scheme is used and that bank **A** has drawn its 20 minutes of allocated time. From that time on, until bank resources are replenished, time is delivered to sessions drawing from bank **A** at low priority.

Also, assume the exact allocation scheme is used and that bank **A** has drawn its $33^1/_3$ minutes. From that time on, time will be delivered to bank **A** at reduced priority. This is, in effect, taking time away from banks **B** and **C**. In other words, time sharing permits competition (over allocation) at reduced priority.

Exclusion from Time Sharing

A bank can be marked as being excluded from time sharing. Marking any bank **A** as being excluded from sharing time from any of its parent banks also excludes that bank's entire sub-tree from sharing time allocated to any of bank **A**'s parent banks.

From the above examples, assume that bank **A** has been excluded from sharing time, that the under allocation scheme is used and that banks **A** and **B** have drawn their allocated 20 minutes. Then, until bank resources are replenished, no resources can be delivered to sessions drawing from bank **A** but sessions drawing from bank **B** can continue to draw resources at low priority.

Temporary Over Allocation

A bank or user allocation may be given a temporary boost in allocations. This temporary allocation reverts to nominal amounts when bank resources are replenished. This feature may be used to temporarily allow a group, project or user access to more time than would normally be the case under special circumstances.

Allocation Carryover

If, when bank resources are being replenished, it is found that a bank or user has not used all of its allocation, a portion of the unused allocation may be "carried over" to the next period. This is similar to the temporary over allocation described above, but is automatic (after being set up by the coordinator) and exists for the purpose of allowing groups who had not been able to use their allocations in a previous period to partially catch up.

Production Prioritization

Production requests drawing from the same bank compete with each other to the extent that the user allocations to those banks are over allocated. (Over allocation of banks from which users are permitted to draw resources is the normal case.) These production requests can be prioritized so that important projects and tasks can get more of the machine's resources than other, less important projects or tasks.

Short Production

Normally, the PWS will schedule production, in priority order, at a rate that keeps the machine busy without causing thrashing. Short production is a feature that allows an organization to declare a production request to be scheduled on demand. That is, short production requests are not subject to the scheduling constraints of other requests. As soon after submission of a short production request that the bank from which it is drawing resources has them available, the request is scheduled to run. However, there is a maximum (configurable) amount of time that short production requests are permitted to run. On systems that have checkpointing facility, short production requests are **not** checkpointable.

Dynamic Reconfigurability

There are several configuration parameters that effectuate policy decisions on the part of the site. These may be changed while the PCS system is running. They are:

priority management

There are six parameters that specify both the normal and standby priority (or "nice") values at which normal production, short production and interactive sessions may run.

maximum allowable memory

To prevent a huge session making CPUs unavailable, there is a parameter that specifies the maximum memory load that can be presented by a process. If a production process reaches this limit, its session is held (in systems that support checkpointing) or killed (in systems that do not).

maximum number of requests

"Queue stuffing" is a perennial problem in most batch systems. People engage in queue stuffing in order to receive a relatively higher share of available resources. The PWS allows queue stuffing, but removes the reason for doing it. This configuration parameter assures that users can not accelerate their rate of resource consumption above a limit determined to be acceptable by the site.

<u>short production request management</u>

Short production introduces a strain on the ability of the PWS to deliver resources as otherwise indicated by normal prioritization means. Therefore, the site that runs PCS can limit the impact of short production by setting the maximum number that can run simultaneously, the maximum number that one user can run simultaneously and the maximum time limit that a short production request can claim after being scheduled.

Actors and roles

The PCS defines or extends prior definitions of various human roles that must be performed to function properly. These roles represent levels of decision making authority. They are:

System Administrator

installs the PCS
creates the RAC data base
creates/destroys any account
modifies any account's attributes
names/removes any account coordinator for any account
permits any bank or user to charge or prohibits any bank or user from charging any account
creates/destroys any bank
allocates/de-allocates resources to/from any bank
names/removes any bank coordinator/designee
creates/destroys/modifies any user allocations to any bank
prioritizes any bank's production load

Account Coordinator

names and removes account coordinators for account
permits any bank of which she or he is also a bank coordinator to charge account
prohibits any bank from charging account
permits any user to charge account or prohibits any user from charging account

Bank Coordinator

creates and destroys sub-banks
names and removes coordinators and designees to bank and sub-banks
allocates and de-allocates resources to or from sub-banks
creates, destroys and modifies user allocations to bank and sub-banks
prioritizes bank's and sub-banks' production load
gives resources away from bank or any sub-bank to any other bank

Bank Coordinator Designee

a bank coordinator, except
- may not create or destroy banks
- may not name or remove coordinators or designees

Supported Environments

The PCS currently runs on UNICOS and Solaris 2.3. We are currently porting it to AIX.

Work in Progress

Massively parallel processing: To support an MPP architecture, Livermore Computing staff is dramatically extending the PCS. We are upgrading PCS to handle a large number of distributed nodes (processing elements) and the notion of node conglomeration (machine partitions). Further we are defining an interface to vendor schedulers (e.g., gang schedulers for time- and space-sharing partitions) to assist in efficiently scheduling the MPP.

Cluster processing: The PCS is being extended to be usable on clusters of workstations as well as in vector-processor and MPP supercomputer environments. To schedule a job on a clustered machine, a user will only need specify the cluster (or a computing feature that only resides on the cluster) rather than any particular node in the cluster.

Heterogeneous computing: The PCS is being further extended to allow cross-host submission of production on any of several heterogeneous platforms in the center from any platform.

Distributed PCS: Support is being added so that allocations and production scheduling is managed for all hosts from a single platform. Coordinators and system administrators will be able to manage the entire PCS system from any host rather than being required to manage it from a single platform.

Future Directions in PCS

Enhanced support of distributed computing: We will be adding support for the execution of a single job on multiple, heterogeneous hosts.

A full complement of job synchronization tools: Support will be added for barriers, events, semaphores, mailboxes and global variables.

Graphical interfaces: Users and administrators will have access to graphical, X-windows interfaces to help make the PCS even easier to use.